

A Mechanistic Dynamic Emulator

C. Albert*

July 6, 2012

Abstract

In applied sciences, we often deal with deterministic simulation models that are too slow for simulation-intensive tasks such as calibration or real-time control. In this paper, an emulator for a generic dynamic model, given by a system of ordinary non-linear differential equations, is developed. The non-linear differential equations are linearized and Gaussian white noise is added to account for the non-linearities. The resulting linear stochastic system is conditioned on a set of solutions of the non-linear equations that have been calculated prior to the emulation. A path-integral approach is used to derive the Gaussian distribution of the emulated solution. The solution reveals that most of the computational burden can be shifted to the conditioning phase of the emulator and the complexity of the actual emulation step only scales like $\mathcal{O}(Nn)$ in multiplications of matrices of the dimension of the state space. Here, N is the number of time-points at which the solution is to be emulated and n the number of solutions the emulator is conditioned on.

The applicability of the algorithm is demonstrated with the hydrological model logSPM.

Keywords: dynamic emulator, path-integral

1 Introduction

In applied sciences, we often have deterministic simulation models at hand that are, although quite accurate, too slow for many simulation-intensive tasks such as calibration or real-time control. The purpose of an *emulator* (e.g. [5]) is a fast interpolation of the response surface of the model. Therefore, the slow deterministic simulation model is simplified and noise is added to account for the errors due to simplification. The resulting fast stochastic model is then conditioned with outputs from the simulation model that have been produced off-line, that is, prior to the actual emulation.

In this paper, we focus on *dynamic* models, i.e., models described by ODE's whose outputs are given by time-series. Treating time as an additional output component or as an additional input [4] and applying a standard Gaussian emulator leads to an emulation time that grows quadratically with the number of time points, which is inefficient if the number of time points is large. Emulators for the time-stepping [2], [3] have the disadvantage that the whole state-space must be emulated if one wants to retain the Markov property of the process. Simplified models in the form of stochastic linear models [8] or linear combinations of (wavelet) basis functions [1] have been used as well. But none of the mentioned approaches uses knowledge

*Eawag, aquatic research, 8600 Dübendorf, Switzerland.

about the dynamics of the simulation model, which we might retrieve, e.g., by linearizing its equations. Recently, Reichert et al. [9] developed a dynamic emulator whose underlying simplified model is a linear stochastic process that captures the first order dynamics of the model. That is, their emulator is to some extent *mechanism-based* and not merely statistical. Furthermore, they applied a Kalman filter in order for the complexity to grow only linearly with the number of time points.

The intention behind this paper is to further improve the computational efficiency of the emulator presented in [9]. I start with a time-continuous linear stochastic process whose drift is assumed to be given by the linearization of the simulation model and whose noise is assumed to account for the non-linearities. This is the same as in [9], except that in [9] the noise is not integrated between time steps. For piece-wise constant input, I derive an analytic solution for the Gaussian distribution describing the emulated output. For this purpose, a path-integral approach seems to be adequate. The analytic solution reveals that most of the computational burden can be shifted to the conditioning phase of the emulator so that we are left with a computational complexity for the emulation step that grows like $\mathcal{O}(Nn)$ in multiplications of matrices of the dimension of the state space, m . Here, N is the number of time points and n the number of simulation outputs on which the stochastic model is conditioned. This is quite a substantial improvement of the algorithm presented in [9], whose emulation step needs $\mathcal{O}(N)$ multiplications of matrices of dimension nm as well as inversions of matrices of dimension m .

Just like in [9], the algorithm is then tested with the hydrological model logSPM.

2 A Generic Dynamic Emulator

Consider a *state space* V of dimension m , whose elements shall be denoted by $\boldsymbol{\xi}$, and a *deterministic simulation model*, given by a system of ordinary differential equations

$$\dot{\boldsymbol{\xi}}(t) = f(\boldsymbol{\xi}(t), \mathbf{x}(t)), \quad (1)$$

where $\mathbf{x} \in W$ denotes inputs and/or parameters and can be time-varying. Subsequently, I refer to \mathbf{x} as input and usually omit its time argument.

The idea behind the emulator is, firstly, to *linearize* eq. (1) and pack all the non-linearities into a *noise* term that is modeled with a standard Wiener process $\boldsymbol{\eta}(t)$ (i.e. Gaussian white noise). The covariance of the noise, C , is assumed to be independent of the input. Thus, the linear stochastic approximation to (1) is given by the system of linear stochastic differential equations

$$\dot{\boldsymbol{\xi}}(t) = A(\mathbf{x})\boldsymbol{\xi}(t) + \mathbf{b}(\mathbf{x}) + C\boldsymbol{\eta}(t). \quad (2)$$

Secondly, $n+1$ *replica of the system are coupled*. Therefore, replace V by $V \otimes \mathbb{R}^{n+1}$ and W by $W \otimes \mathbb{R}^{n+1}$. Henceforth, vectors without indices will denote elements of these extended spaces. The $n+1$ replica are associated with $n+1$ different inputs, \mathbf{x}^α . The first n inputs are those for which solutions of (1) are calculated that will be used for the conditioning while the $(n+1)$ th input is the one for which a solution is to be emulated. The replica are assumed to couple through the noise term only. Thus, $A(\mathbf{x})$ and $\mathbf{b}(\mathbf{x})$ now denote the tensors

$$A_\beta^\alpha(\mathbf{x}) = A(\mathbf{x}^\alpha)\delta_\beta^\alpha, \quad (3)$$

$$\mathbf{b}^\alpha(\mathbf{x}) = \mathbf{b}(\mathbf{x}^\alpha). \quad (4)$$

The closer the inputs (measured with some metric ρ on W) the stronger the associated replica are assumed to couple. Hence, set

$$\tilde{C}(\mathbf{x}) = C \otimes R(\mathbf{x}),$$

with

$$R^{\alpha\beta}(\mathbf{x}) = \exp(-\rho(\mathbf{x}^\alpha, \mathbf{x}^\beta)/2).$$

Thus, the emulator is described by the nm coupled linear stochastic differential equations

$$\dot{\boldsymbol{\xi}}(t) = A(\mathbf{x})\boldsymbol{\xi}(t) + \mathbf{b}(\mathbf{x}) + \tilde{C}(\mathbf{x})\boldsymbol{\eta}(t). \quad (5)$$

Next, I derive the probability density of $\boldsymbol{\xi}(t)$ on the space of paths $[t_0, t_N] \longrightarrow V \otimes \mathbb{R}^{n+1}$, with initial condition

$$\boldsymbol{\xi}(t_0) = 0. \quad (6)$$

It reads

$$\begin{aligned} P[\boldsymbol{\xi}(t)] &\propto \exp \left[-\frac{1}{2} \int_{t_0}^{t_N} \left(\dot{\boldsymbol{\xi}}(t) - A(\mathbf{x})\boldsymbol{\xi}(t) - \mathbf{b}(\mathbf{x}) \right)^\dagger (\tilde{C}\tilde{C}^T)^{-1}(\mathbf{x}) \left(\dot{\boldsymbol{\xi}}(t) - A(\mathbf{x})\boldsymbol{\xi}(t) - \mathbf{b}(\mathbf{x}) \right) dt \right] \\ &= \exp \left[-\frac{1}{2} \int_{t_0}^{t_N} (\boldsymbol{\xi}(t) - D^{-1}\mathbf{b}(\mathbf{x}))^\dagger (D^\dagger(\tilde{C}\tilde{C}^T)^{-1}D)(\mathbf{x}) (\boldsymbol{\xi}(t) - D^{-1}\mathbf{b}(\mathbf{x})) dt \right], \end{aligned} \quad (7)$$

where

$$D = \frac{\partial}{\partial t} - A(\mathbf{x}). \quad (8)$$

To proceed, I need to determine the *Green's function* of D . The most general solution of

$$D(t)G(t, t') = \delta(t - t') \quad (9)$$

is given by (see, e.g., [6] Chapter 3.3)

$$G(t, t') = (\bar{\Theta}(t - t') + c(t'))\mathcal{P} \exp \left[\int_{t'}^t A(\mathbf{x}(\tau)) d\tau \right], \quad (10)$$

where $\bar{\Theta}(t - t')$ is the regularized Heavyside function with $\bar{\Theta}(0) = 1/2$ and \mathcal{P} denotes path-ordering of the exponential. The function $c(t')$ is determined by the boundary condition (6), which entails

$$G(t_0, t) = 0,$$

and translates into

$$c(t') \equiv 0.$$

Then, the adjoint Green's function reads as

$$G^\dagger(t, t') = \bar{\Theta}(t' - t)\mathcal{P} \exp \left[-\int_{t'}^t A^T(\mathbf{x}(\tau)) d\tau \right]. \quad (11)$$

Now, I calculate the *correlation functions* for two replica at two different time points, as expressed by the $m \times m$ matrices

$$\tilde{\Sigma}_{ij}^{\alpha\beta} = \langle \boldsymbol{\xi}^\alpha(t_i) \otimes \boldsymbol{\xi}^\beta(t_j) \rangle = Z^{-1} \int \exp \left[-\frac{1}{2} \boldsymbol{\xi}^\dagger(t) D^\dagger(\tilde{C}\tilde{C}^T)^{-1} D(\mathbf{x}) \boldsymbol{\xi}(t) \right] \boldsymbol{\xi}^\alpha(t_i) \otimes \boldsymbol{\xi}^\beta(t_j) \mathcal{D}\boldsymbol{\xi}, \quad (12)$$

with

$$Z = \int \exp \left[-\frac{1}{2} \boldsymbol{\xi}^\dagger(t) D^\dagger (\tilde{C} \tilde{C}^T)^{-1} D(\mathbf{x}) \boldsymbol{\xi}(t) \right] \mathcal{D}\boldsymbol{\xi}.$$

Using (10) and (11) one finds that, for $t_i \geq t_j$,

$$\tilde{\Sigma}_{ij}^{\alpha\beta} = \int_{t_0}^{t_j} \mathcal{P} \exp \left[\int_{t'}^{t_i} A(\mathbf{x}^\alpha(\tau)) d\tau \right] (\tilde{C} \tilde{C}^T)^{\alpha\beta}(\mathbf{x}(t')) \mathcal{P} \exp \left[- \int_{t_j}^{t'} A^T(\mathbf{x}^\beta(\tau)) d\tau \right] dt'. \quad (13)$$

For $t_i < t_j$, one may use the symmetry relations

$$\tilde{\Sigma}_{ij}^{\alpha\beta} = (\tilde{\Sigma}_{ji}^{\beta\alpha})^T. \quad (14)$$

All *finite dimensional marginals* of (7) will be Gaussians. I consider the finite-dimensional subspace of those components of the first n replica that are simulated with (1) and those components of the $(n+1)$ th replica that shall be emulated, both at time points $t_0 < t_1 < \dots < t_N$. Therefore, I introduce the operator $H(\mathbf{x})$ that is defined as

$$(H(\mathbf{x})[\boldsymbol{\xi}(t)])_i^\alpha = H(\mathbf{x}^\alpha(t_i)) \boldsymbol{\xi}^\alpha(t_i), \quad (15)$$

where, on the r.h.s., $H(\mathbf{x}^\alpha(t_i)) =: H_i^\alpha$ denotes matrices of constant rank $m' < m$. The image of (15) is supposed to be determined by eq.

$$(H(\mathbf{x})[\boldsymbol{\xi}(t)])_i^\alpha = \mathbf{y}_i^\alpha. \quad (16)$$

Integrating out all degrees of freedom that are not determined by (16) yields the Gaussian distribution¹

$$\begin{aligned} Z^{-1} \int P[\boldsymbol{\xi}(t)] \delta(H[\boldsymbol{\xi}(t)] - \mathbf{y}) \mathcal{D}\boldsymbol{\xi} \\ \propto \int \exp \left[-\frac{1}{2} \int \boldsymbol{\xi}^\dagger(t) D^\dagger (\tilde{C} \tilde{C}^T)^{-1} D \boldsymbol{\xi}(t) dt \right] \delta(H[\boldsymbol{\xi}(t)] - (\mathbf{y} - HD^{-1}\mathbf{b})) \mathcal{D}\boldsymbol{\xi} \\ \propto \exp \left[-\frac{1}{2} (\mathbf{y} - HD^{-1}\mathbf{b})^\dagger \Sigma^{-1} (\mathbf{y} - HD^{-1}\mathbf{b}) \right], \end{aligned} \quad (17)$$

with covariance matrix

$$\Sigma = HD^{-1}(\tilde{C} \tilde{C}^T)(D^\dagger)^{-1}H^\dagger, \quad (18)$$

and mean

$$\mathbf{z} = HD^{-1}\mathbf{b}. \quad (19)$$

The covariance matrix is a square matrix of dimension $(n+1)Nm'$, whose $m' \times m'$ blocks are given by the equations

$$\Sigma_{ij}^{\alpha\beta} = H_i^\alpha \tilde{\Sigma}_{ij}^{\alpha\beta} (H_j^\beta)^T, \quad (20)$$

with $\tilde{\Sigma}$ as defined by (13) and (14).

Finally, I determine the *Gaussian distribution for the $(n+1)$ th replica (the online system)*, conditioned on the simulations \mathbf{y}_i^a , for $i = 1, \dots, N$ and $a = 1, \dots, n$. Therefore, I split the $(n+1)$ th replica off, writing Σ as the block matrix

$$\Sigma = \left(\begin{array}{c|c} \Sigma^{n+1,n+1} & \Sigma^{n+1,.} \\ \hline \Sigma^{.,n+1} & \Sigma' \end{array} \right). \quad (21)$$

¹To keep the notation simple, we will omit the dependence of H , D , \tilde{C} , and \mathbf{b} on \mathbf{x} subsequently.

Mean and covariance matrix of the online system are given by equations

$$\bar{\mathbf{y}} = \mathbf{z}^{n+1} + \Sigma^{n+1,a}(\Sigma')_{ab}^{-1}(\mathbf{y}^b - \mathbf{z}^b), \quad (22)$$

$$\bar{\Sigma} = \Sigma^{n+1,n+1} - \Sigma^{n+1,a}(\Sigma')_{ab}^{-1}\Sigma^{b,n+1}, \quad (23)$$

where a and b run from 1 to n only. Note that eq. (22) is translational invariant. Thus, one may replace condition (6) by

$$\boldsymbol{\xi}(0) = \boldsymbol{\xi}_0. \quad (24)$$

In the remainder of this chapter, a recursive procedure of calculating (22) and (23) is developed. Due to path-ordering eqs. (13) and (20) can be written as

$$\Sigma_{ij}^{\alpha\beta} = H_i^\alpha \left(\sum_{k=0}^{j-1} h_{i-1}^\alpha \dots h_{k+1}^{\alpha\beta} g_k^{\alpha\beta} h_{k+1}^{\dagger\beta} \dots h_{j-1}^{\dagger\beta} \right) (H_j^\beta)^T, \quad (25)$$

with

$$g_k^{\alpha\beta} = \int_{t_k}^{t_{k+1}} \mathcal{P} \exp \left[\int_{t'}^{t_{k+1}} A(\mathbf{x}^\alpha(\tau)) d\tau \right] (\tilde{C}\tilde{C}^T)^{\alpha\beta}(t') \mathcal{P} \exp \left[- \int_{t_{k+1}}^{t'} A^T(\mathbf{x}^\beta(\tau)) d\tau \right], \quad (26)$$

$$h_l^\alpha = \mathcal{P} \exp \left[\int_{t_l}^{t_{l+1}} A(\mathbf{x}^\alpha(\tau)) d\tau \right], \quad (27)$$

$$h_l^{\dagger\alpha} = \mathcal{P} \exp \left[- \int_{t_{l+1}}^{t_l} A^T(\mathbf{x}^\alpha(\tau)) d\tau \right]. \quad (28)$$

The boundary conditions (6) or (24) imply the initial variances

$$\Sigma_{00}^{\alpha\beta} = 0. \quad (29)$$

In the *conditioning step* of the algorithm one calculates $(\Sigma')^{-1}$ and \mathbf{z}^a , for $a = 1, \dots, n$. For the former, use (14), (20) and (29) and, for $j \leq i$, the recursion relations

$$\Sigma_{i+1,j}^{\alpha\beta} = h_i^\alpha \Sigma_{ij}^{\alpha\beta} \quad (30)$$

$$\Sigma_{ii}^{\alpha\beta} = \Sigma_{i,i-1}^{\alpha\beta} h_{i-1}^{\dagger\beta} + g_{i-1}^{\alpha\beta}. \quad (31)$$

For the latter, set

$$\mathbf{z}_i^\alpha = H_i^\alpha \tilde{\mathbf{z}}_i^\alpha,$$

and use the recursion relations

$$\tilde{\mathbf{z}}_{i+1}^\alpha = h_i^\alpha \tilde{\mathbf{z}}_i^\alpha + \mathbf{k}_i^\alpha, \quad \tilde{\mathbf{z}}_0^\alpha = 0, \quad (32)$$

with

$$\mathbf{k}_i^\alpha = \int_{t_i}^{t_{i+1}} \mathcal{P} \exp \left[\int_{t'}^{t_{i+1}} A(\mathbf{x}^\alpha(\tau)) d\tau \right] \mathbf{b}^\alpha(t') dt'. \quad (33)$$

Once $(\Sigma')^{-1}$ and all the \mathbf{z}^a are calculated, pre-calculate, for the emulation step, the covectors

$$\mathbf{z}'_{ia} := T_{ij}^a (H_j^a)^T ((\Sigma')^{-1})_{ab}^{jk} (\mathbf{y}_k^b - \mathbf{z}_k^b), \quad (34)$$

where, on the r.h.s., the i 's and the a 's are not summed over and with

$$T_{ij}^a := \begin{cases} h_{i+1}^{\dagger a} \dots h_{j-1}^{\dagger a}, & j \geq i+2, \\ \mathbf{1}, & j = i+1, \\ 0, & \text{else.} \end{cases} \quad (35)$$

In the actual *emulation step* calculate (22) setting

$$\bar{\mathbf{y}}_i = H_i^{n+1} \tilde{\mathbf{y}}_i, \quad (36)$$

and using the recursion relation

$$\tilde{\mathbf{y}}_{i+1} = h_i^{n+1} \tilde{\mathbf{y}}_i + \mathbf{k}_i^{n+1} + g_i^{n+1,a} \mathbf{z}'_{ia}, \quad (37)$$

with \mathbf{z}'_{ia} as defined in (34). In order to get the start value, $\tilde{\mathbf{y}}_1$, one needs to calculate $\Sigma_{1j}^{n+1,a}$ using (14), (29) and the recursion relations (30) and (31). The computational complexity of the emulation step is of the order $\mathcal{O}(Nn)$ in matrix multiplications of dimension m . If one is interested in the variances, i.e., the diagonal elements of $\bar{\Sigma}$, one may derive a similar recursion formula for them.

Since path-ordered exponentials can, in general, not be calculated analytically, I consider the special case of *piece-wise constant input*

$$\mathbf{x}^\alpha(t) = \mathbf{x}_i^\alpha, \quad t_i \leq t \leq t_{i+1}.$$

Then, (26), (27) and (33) reduce to

$$g_k^{\alpha\beta} = (R_k^{\alpha\beta})^2 \int_{t_k}^{t_{k+1}} e^{(t_{k+1}-t')A(\mathbf{x}_k^\alpha)} C C^T e^{(t_{k+1}-t')(A(\mathbf{x}_k^\beta))^T} dt', \quad (38)$$

$$h_l^\alpha = e^{(t_{l+1}-t_l)A(\mathbf{x}_l^\alpha)}, \quad (39)$$

$$\mathbf{k}_i^\alpha = \int_{t_i}^{t_{i+1}} e^{(t_{i+1}-t')A(\mathbf{x}_i^\alpha)} \mathbf{b}_i^\alpha dt'. \quad (40)$$

If $A(\mathbf{x})$ is *diagonalizable*, functions (38) through (40) can be obtained analytically. For

$$A(\mathbf{x}_k^\alpha) = M_k^\alpha \text{diag}_o[\lambda_o^\alpha] (M_k^\alpha)^{-1},$$

one gets

$$g_k^{\alpha\beta} = (R_k^{\alpha\beta})^2 M_k^\alpha B_k^{\alpha\beta} (M_k^\beta)^T, \quad (41)$$

with

$$(B_k^{\alpha\beta})^p_q = \frac{\exp((t_{k+1}-t_k)(\lambda_{k,p}^\alpha + \lambda_{k,q}^\beta)) - 1}{\lambda_{k,p}^\alpha + \lambda_{k,q}^\beta} ((M_k^\alpha)^{-1} C C^T ((M_k^\beta)^{-1})^T)^p_q, \quad (42)$$

and

$$h_l^\alpha = M_l^\alpha \text{diag}_o[\exp((t_{l+1}-t_l)\lambda_{l,o}^\alpha)] (M_l^\alpha)^{-1}, \quad (43)$$

and

$$\mathbf{k}_i^\alpha = M_i^\alpha \text{diag}_o \left[\frac{\exp((t_{i+1}-t_i)\lambda_{i,o}^\alpha) - 1}{\lambda_{i,o}^\alpha} \right] (M_i^\alpha)^{-1} \mathbf{b}_i^\alpha. \quad (44)$$

If \mathbf{x} is time-independent (e.g. parameters of the model) and $A(\mathbf{x})$ diagonalizable, (20) can be calculated explicitly. If

$$A(\mathbf{x}^\alpha) = M^\alpha \text{diag}_o[\lambda_o^\alpha] (M^\alpha)^{-1},$$

one derives from (20) that, for $t_i > t_j$,

$$\tilde{\Sigma}_{ij}^{\alpha\beta} = (R^{\alpha\beta})^2 M^\alpha B_{ij}^{\alpha\beta} (M^\beta)^T,$$

where

$$\begin{aligned} (B_{ij}^{\alpha\beta})^p_q &= ((M^\alpha)^{-1} C C^T ((M^\beta)^T)^{-1})^p_q \int_{t_0}^{t_j} \exp[t_i \lambda_p^\alpha + t_j \lambda_q^\beta - t'(\lambda_p^\alpha + \lambda_q^\beta)] dt' \\ &= ((M^\alpha)^{-1} C C^T ((M^\beta)^T)^{-1})^p_q \frac{\exp\left((t_i - t_0)\lambda_p^\alpha + (t_j - t_0)\lambda_q^\beta\right) - \exp\left((t_i - t_j)\lambda_p^\alpha\right)}{\lambda_p^\alpha + \lambda_q^\beta}. \end{aligned} \quad (45)$$

3 Hydrological Application

In this section, the algorithm developed in the last section is tested with a simple hydrological model called logSPM [7]. The state vector of this model is three-dimensional,

$$\boldsymbol{\xi} = (h_s, h_{gw}, h_r)^T, \quad (46)$$

and describes the amount of water stored in the soil, the ground-water and the river. The dynamics is described by the system of ordinary differential equations

$$\dot{h}_s = q_{rain} - q_{runoff} - q_{et} - q_{lat} - q_{gw}, \quad (47)$$

$$\dot{h}_{gw} = q_{gw} - q_{bf} - q_{dp}, \quad (48)$$

$$\dot{h}_r = q_{runoff} + q_{lat} + q_{bf} - q_r, \quad (49)$$

and visualized in Fig. 1. The fluxes are given by the equations

$$q_{rain} = i_{rain}(t), \quad q_{runoff} = f_{sat} i_{rain}(t), \quad (50)$$

$$q_{et} = f_{et} i_{pet}(t), \quad q_{lat} = f_{sat} q_{lat,max}, \quad (51)$$

$$q_{gw} = f_{sat} q_{gw,max}, \quad q_{bf} = k_{bf} h_{gw}, \quad (52)$$

$$q_{dp} = k_{dp} h_{gw}, \quad q_r = k_r h_r, \quad (53)$$

with the fraction of saturated area, f_{sat} , given by equation

$$f_{sat} = \frac{1}{1 + s_F e^{-k_s h_s}} - \frac{1}{1 + s_F}, \quad (54)$$

and the fraction of actual evapotranspiration, f_{et} , given by equation

$$f_{et} = 1 - e^{-k_{et} h_s}. \quad (55)$$

The output of the model is the river flow, Q_r , given as

$$Q_r = A_W q_r,$$

where A_W is the area of watershed.

The linearization of the model equations reads:

$$\dot{\boldsymbol{\xi}}(t) = A(\mathbf{x})\boldsymbol{\xi}(t) + \mathbf{b}(\mathbf{x}), \quad (56)$$

with

$$A(\mathbf{x}) = \begin{pmatrix} \lambda_1(t) & 0 & 0 \\ a & \lambda_2 & 0 \\ c(t) & b & \lambda_3 \end{pmatrix}, \quad \mathbf{b}(\mathbf{x}) = \begin{pmatrix} i_{rain}(t) \\ 0 \\ 0 \end{pmatrix}, \quad (57)$$

and

$$\mathbf{x}(t) = (\lambda_1(t), \lambda_2, \lambda_3, a, b, c(t), i_{rain}(t))^T,$$

with

$$a = a_{sat}q_{gw,max}, \quad b = k_{bf}, \quad c(t) = a_{sat}(i_{rain}(t) + q_{lat,max}), \quad (58)$$

and

$$\lambda_1(t) = -a_{sat}(i_{rain}(t) + q_{lat,max} + q_{gw,max}) - a_{et}i_{pet}(t), \quad \lambda_2 = -k_{bf} - k_{dp}, \quad \lambda_3 = -k_r. \quad (59)$$

The functions (54) and (55) were approximated by linear functions that intersect the nonlinear functions at $h_{s,1}$ and $h_{s,2}$, respectively. See Fig. 2. Therefore,

$$a_{sat} = \frac{1}{h_{s,1}} \left(\frac{1}{1 + s_F e^{-k_{sg}h_{s,1}}} - \frac{1}{1 + s_F} \right),$$

and

$$a_{et} = \frac{1}{h_{s,2}} \left(1 - e^{-k_{et}h_{s,2}} \right).$$

Only the inputs i_{rain} and i_{pet} are time-dependent, and, therefore, c , \mathbf{b} and λ_1 . The observation matrices read as

$$H^\alpha = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -A_W \lambda_3^\alpha \end{pmatrix}. \quad (60)$$

I choose the Euclidean metric on the space of parameters

$$\mathbb{R}^8 \ni \boldsymbol{\theta} = (k_s, s_F, k_{et}, q_{lat,max}, q_{gw,max}, k_{bf}, k_{dp}, k_r)^T,$$

where each component is normalized with a reasonable range. The noise C was chosen to be diagonal and the diagonal entries to be a certain fraction of the initial condition $\boldsymbol{\xi}_0$.

Obviously, $A(\mathbf{x})$ is diagonalizable:

$$M^{-1}(t)A(\mathbf{x})M(t) = \text{diag}_o[\lambda_o], \quad (61)$$

with

$$M(t) = \begin{pmatrix} 1 & 0 & 0 \\ \frac{a}{\lambda_1 - \lambda_2} & 1 & 0 \\ \frac{c(\lambda_1 - \lambda_2) + ab}{(\lambda_1 - \lambda_2)(\lambda_1 - \lambda_3)} & \frac{b}{\lambda_2 - \lambda_3} & 1 \end{pmatrix}, \quad (62)$$

and the matrices (41), (43) and (44) can be calculated analytically. Plot 3 compares solutions of the full model with emulated solutions for 5 randomly chosen sets of parameters (that were not used for the conditioning of the emulator). The results are very similar to those obtained in [9]. For an extended statistical analysis of the performance of the emulator, I refer to [9].

4 Conclusions

I have presented an explicit solution for the emulation of the time-series of a dynamic model. In general, the path-ordered exponentials the solution is expressed with cannot be calculated analytically. Therefore, I resort to piece-wise constant inputs. Then, the emulator presented in this paper is the same as the one presented in [9], except that I integrate the noise between time steps. For piece-wise constant input, this can be done at negligible additional cost and potentially increases the quality of the emulation.

The exact solution presented in eqs. (22) and (23) allows for an efficient numerical implementation that is of the order $\mathcal{O}(nN)$ in matrix multiplications of dimension m . The Kalman filtering and smoothing algorithm used in [9] needs $\mathcal{O}(N)$ matrix multiplications of dimension nm and matrix inversions of dimension m .

The disadvantage of my method, however, as compared to the one presented in [9] is the fact that a huge matrix of dimension Nnm' needs to be inverted for the conditioning, which might be challenging both for the memory and the CPU.

Acknowledgments:

I'm indebted to Peter Reichert for many fruitful discussions about emulators as well as many lines of R Code for the presentation of the results.

References

- [1] M. J. Bayarri, J. O. Berger, J. Cafeo, G. Garcia-Donato, F. Liu, J. Palomo, R. J. Parthasarathy, R. Paulo, J. Sacks, and D. Walsh. Computer model validation with functional output. *Ann. Stat.*, 35(5):1874–1906, 2007.
- [2] S. Bhattacharya. A simulation approach to bayesian emulation of complex dynamic computer models. *Bayesian Analysis*, 2:783–816, 2007.
- [3] S. Conti, J. P. Gosling, J. Oakley, and A. O'Hagan. Gaussian process emulation of dynamic computer codes. *Biometrika*, 96(3):663–676, 2009.
- [4] S. Conti and A. O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. *J. Stat. Planning and Inference*, 140:640–651, 2010.
- [5] M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *J. Roy. Stat. Soc. B*, 63(3):425–464, 2001.
- [6] H. Kleinert. *Path integrals in quantum mechanics, statistics, polymer physics, and financial markets (5th ed.)*. World Scientific Publishing, Singapore, 2009.
- [7] G. Kuczera, D. Kavetski, S. Franks, and M. Thyer. Towards a Bayesian total error analysis of conceptual rainfall-runoff models: Characterising model error using storm-dependent parameters. *J. Hydrology*, 331(1-2):161–177, 2006.
- [8] F. Liu and M. West. A dynamic modelling strategy for Bayesian computer model emulation. *J. Bayesian Anal.*, 4(2):393–412, 2009.

- [9] P. Reichert, G. White, M. J. Bayarri, and E. B. Pitman. Mechanism-based emulation of dynamic simulation models: Concept and application in hydrology. *J. Comp. Stat. Dat. Anal.*, 55:1638–1655, 2011.

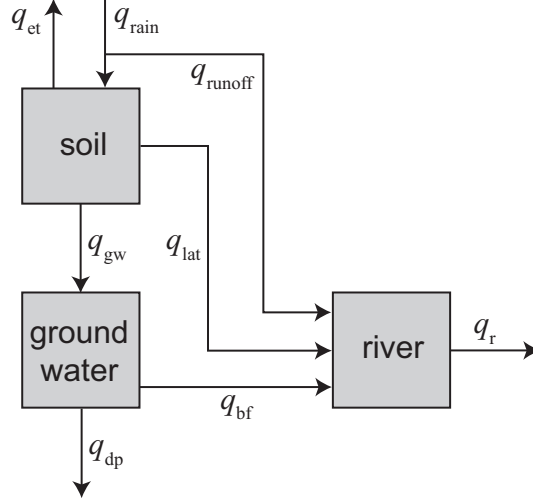


Figure 1: Visualization of the fluxes in the model logSPM. Taken from J. Comp. Stat. and Data Analysis.

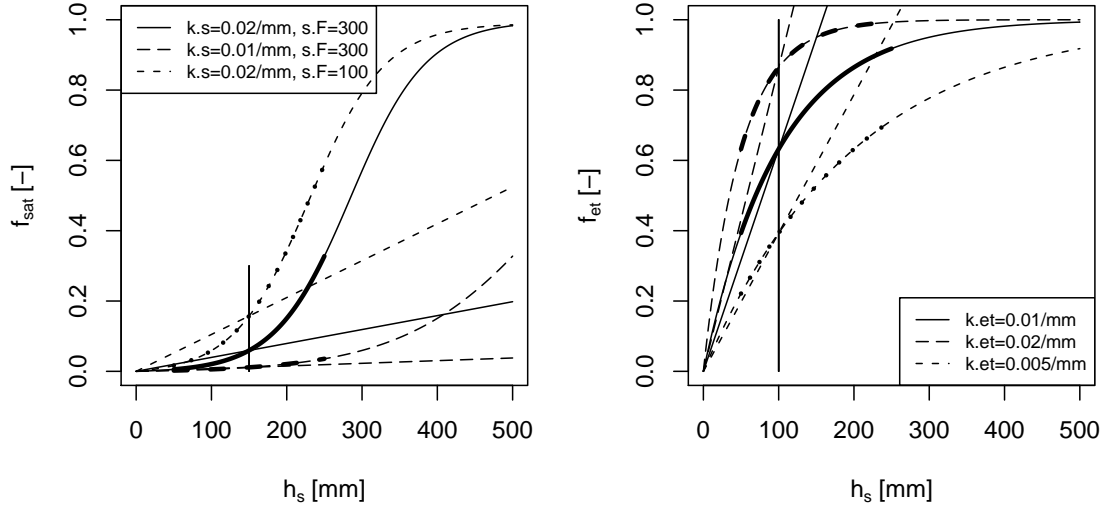


Figure 2: Shape of the nonlinear functions used for describing some fluxes in the hydrological model. Fraction of saturated are, left, and fraction of actual evapotranspiration, right. The bold parts of the curves represent the range of values covered in the base simulation. The straight lines represent linearizations that intersect the nonlinear function at given values of h_s . Taken from J. Comp. Stat. and Data Analysis.

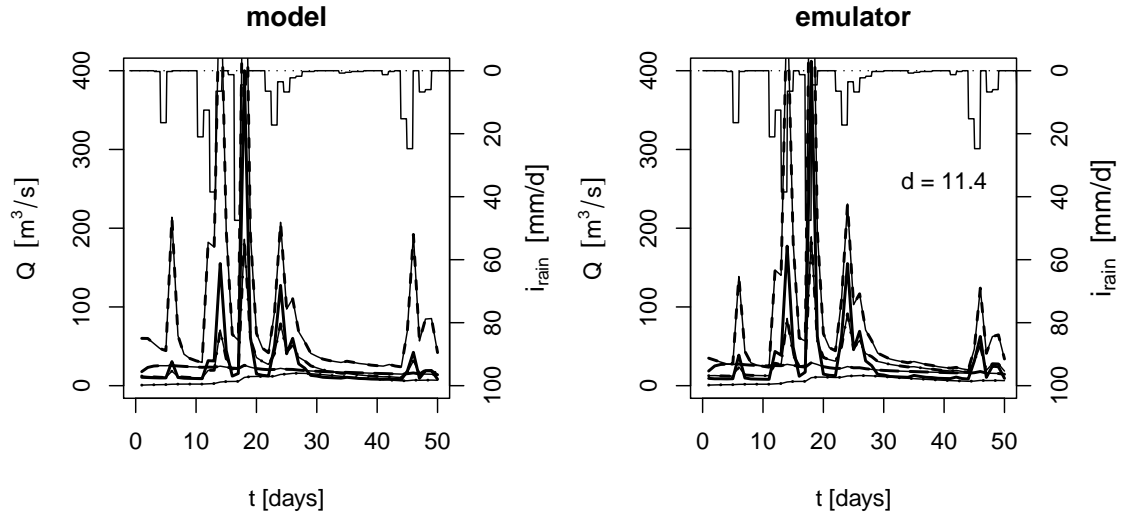


Figure 3: Comparison of simulations of the full model (left) with emulations (right) for 5 randomly chosen sets of parameters. The dominant model input, rain intensity, is plotted from the top (right scale). The emulator was conditioned with 50 sets of parameters. The d -value is the square root of the mean sum of squares.